Images, Strings

CSE 120 Winter 2020

Instructor: Sam Wolfson

Teaching Assistants: Yae Kubota, Eunia Lee, Erika Wolfe

The 2020 Election Will Be a War of Disinformation

“Every presidential campaign sees its share of spin and misdirection, but this year’s contest promises to be different. In conversations with political strategists and other experts, a dystopian picture of the general election comes into view—one shaped by coordinated bot attacks, Potemkin local-news sites, micro-targeted fearmongering, and anonymous mass texting. Both parties will have these tools at their disposal. But in the hands of a president who lies constantly, who traffics in conspiracy theories, and who readily manipulates the levers of government for his own gain, their potential to wreak havoc is enormous.”

- https://www.theatlantic.com/magazine/archive/2020/03/the-2020-disinformation-war/605530/

Administrivia

- Assignments:
  - Arrays and Elli [checkoff] due Friday (2/14)
    - Recommend getting checked off by the end of section on Thursday
  - Color Filters [checkoff] due Tuesday (2/18)
  - Word Guessing [checkoff] due Tuesday (2/18)
- Quiz 3 this Friday
  - Topics and snippets posted on website
  - We'll drop your lowest quiz
- Big Ideas: Artificial Intelligence
  - Reading Check 6 due Thursday (2/13) before section

Outline

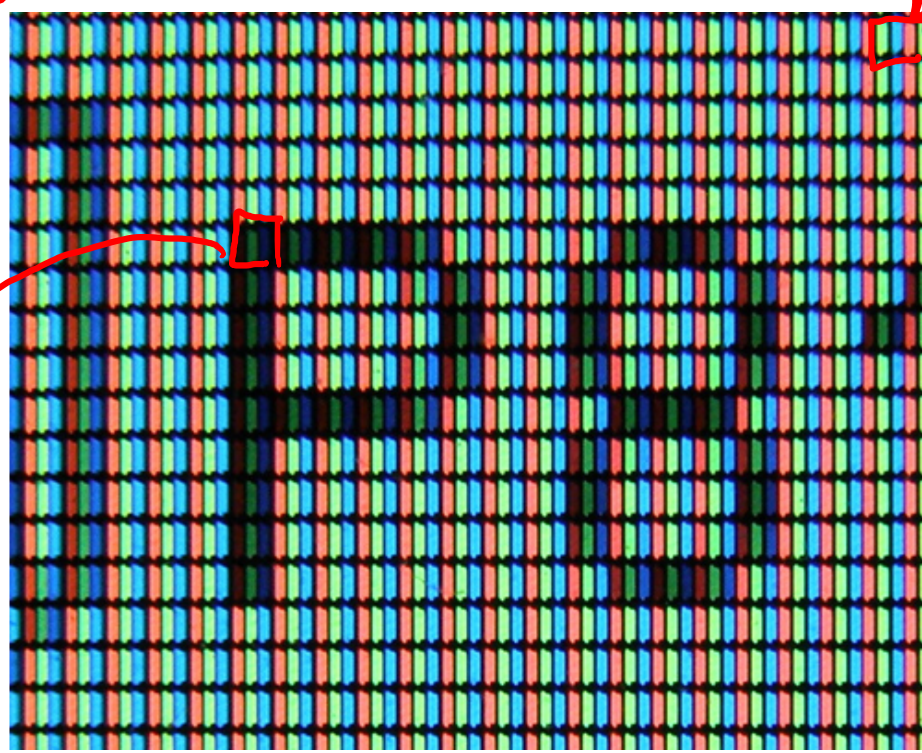
- **Images**
- Compression
- Strings

Images

- An image is just a 2-dimensional set of pixels
  - The image has a *width* and a *height*
  - Each pixel has an associated (RGB) *color*

width

white



black

height

Images

- An image is just a 2-dimensional set of pixels
  - The image has a *width* and a *height*
  - Each pixel has an associated (RGB) *color*
- In Processing, an image is represented as an array of `color` data
  - Can explicitly use `color[] myImage`
  - Processing also provides special datatype `PImage`

Using Images in Processing

my-proj:
- my_proj.pde
- photo.jpg

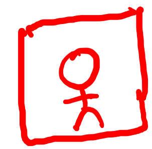
1) Load an image from a file into a Processing variable
   - Use the `loadImage("photo.jpg")` function
     - The image name is a String representing the *path* to the file, similar to your website
   - Store the return value from `loadImage()` into a `PImage` variable
     - *e.g.* `PImage myImg = loadImage("img/sam.jpg");`

2) Draw the image on your canvas using the `image()` function

   top-left
   - `image(<PImage var>, <x>, <y>)`
   - *e.g.* `image(myImg, 0, 0);`

The Canvas as an Image

- The drawing canvas itself is also treated as an image!
  - Retrieve the current canvas image data (*i.e.* array of `color` data) using the `loadPixels()` function
    - `loadPixels()` has no parameters or return value
    - The canvas image data will be automatically stored into the system variable `pixels[]`



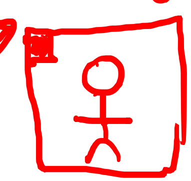
pixels[]

pixels[0] = color(0)

  - You can manually manipulate the data in `pixels[]`
    - *e.g.* `pixels[0] = color(0);  // set to black`

pixels[]

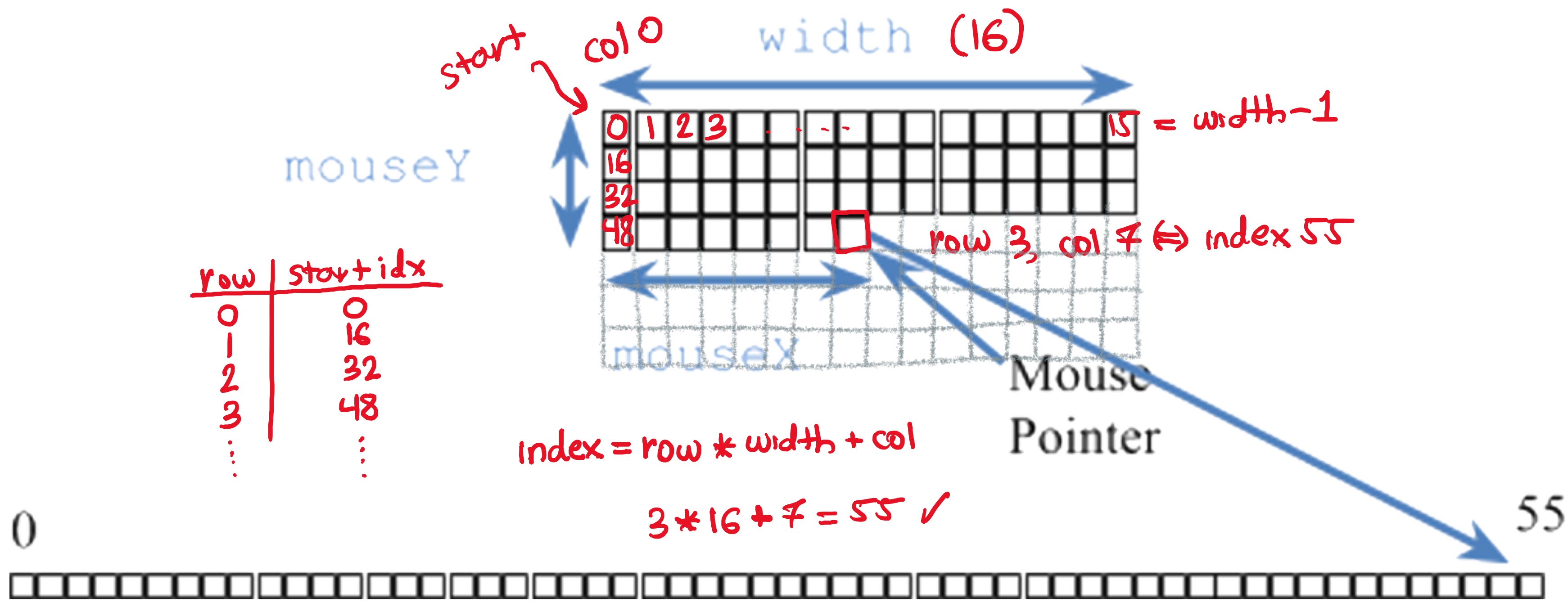
  - Update the drawing canvas with the current/new data in `pixels[]` using the `updatePixels()` function
    - `updatePixels()` also has no parameters or return value



black

Linearizing an Image

- Despite being 2-D in nature (*i.e.* x- and y-coordinates), we deal with image data in a 1-D array (*i.e.* `pixels[]`) length n has indices 0 to n-1
  - As we increment our array index, we move left-to-right horizontally and then top-to-bottom vertically



row	start idx
0	0
1	16
2	32
3	48
⋮	⋮

index = row * width + col

3 * 16 + 7 = 55 ✓

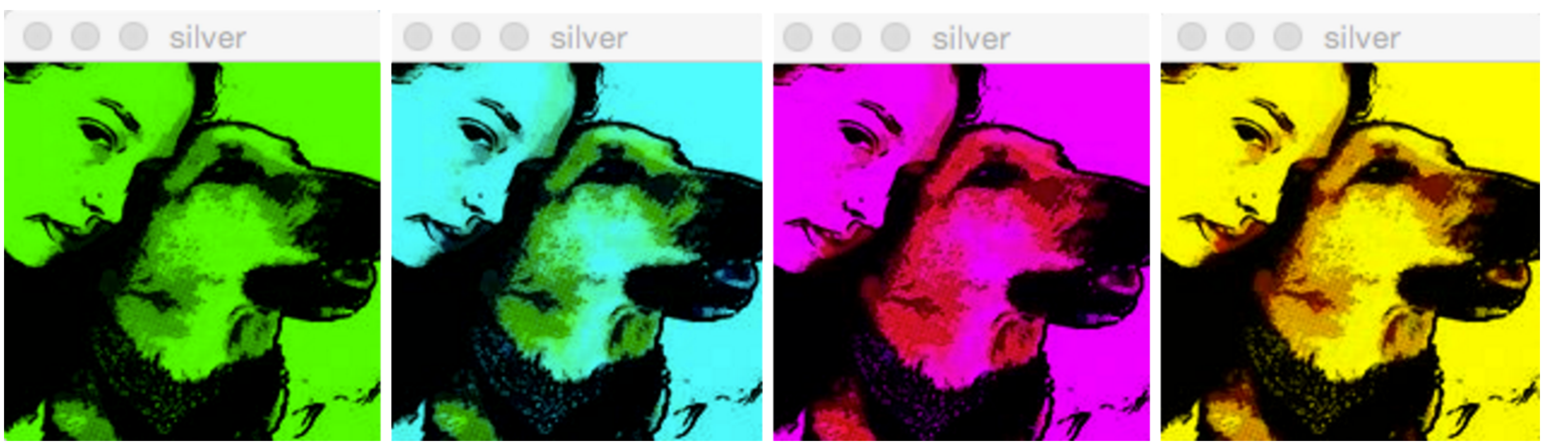
Color as Data in Processing

- **Recall:** all data on a computer is stored using *binary encoding*
  - Including colors, though we won't cover exactly how

- Processing has a special `color` datatype
  - We're used to using the `color(R, G, B)` function to specify colors
  - Represents colors but looks nonsensical if you try to print it
  - Can retrieve the RGB triplet values using the functions `red()`, `green()`, and `blue()`

Color Filters

- Learn the basics of using and manipulating images in Processing
  - You choose a photo to display
  - Display the RGB of the pixel your mouse is hovering over
  - Key presses will filter the colors of your image appropriately



Outline

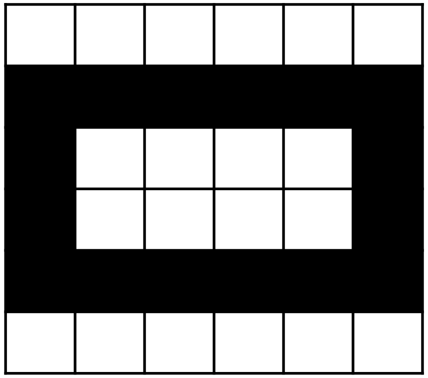
- Images
- **Compression**
- Strings

Compression

- Compression is the process of encoding information/data using fewer bits than the original representation
  - Lossless: original bits can be *exactly* recovered from transformed bits
  - Lossy: original bits *cannot* be exactly recovered from transformed bits (*i.e.* some data is lost)

Lossless Compression

- Eliminates bits that can be recovered again
- Consider this 6 x 6 black-and-white image:

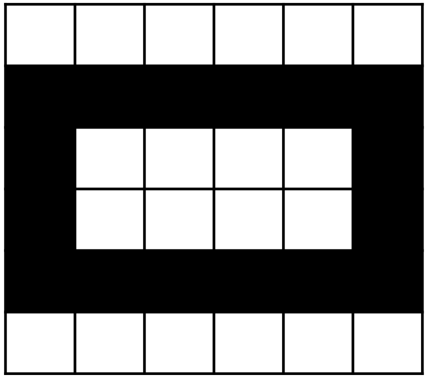


- Uncompressed:
  - WWWWWW BBBBBB BWWWWB BWWWWB BBBBBB WWWWWW

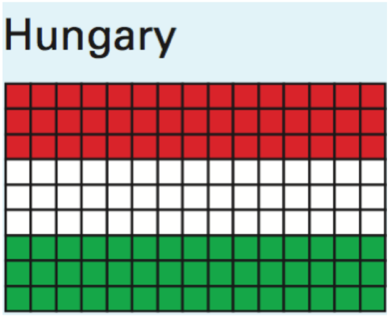
6W 6W

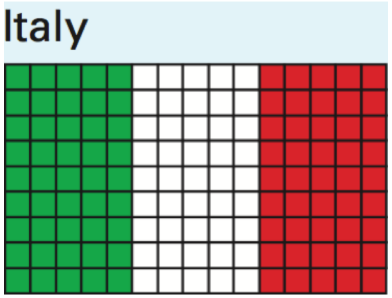
Lossless Image Format: RLE

- Run Length Encoding
  - Not used commonly, but found in formats (TIFF and Bitmap)
  - For repeated data/color, encode # of repeats
  - Many variations on actual encoding exist
- Black-and-white example:
  - `6W 7B 4W 2B 4W 7B 6W`



- Flag example:
  - HU = 45:R,45:W,45:G
  - IT = 5:G,5:W,5:R,5:G,5:W,5:R
    5:G,5:W,5:R,5:G,5:W,5:R
    5:G,5:W,5:R,5:G,5:W,5:R
    5:G,5:W,5:R,5:G,5:W,5:R5:G,5:W,5:R



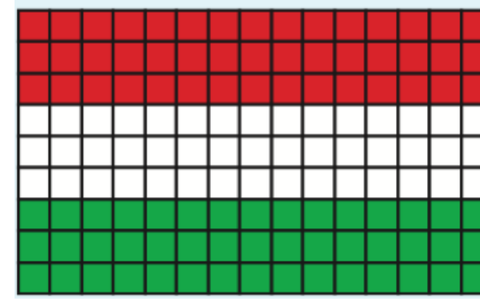


Lossless Image Format: GIF, PNG

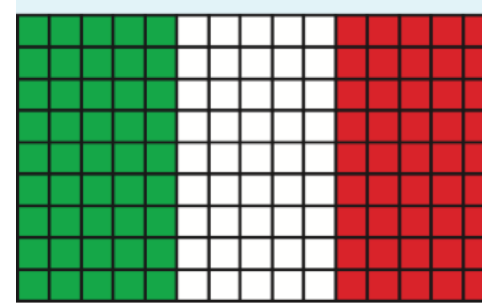
- Graphics Interchange Format
  - Uses a 256-color palette (not RGB) encoded in a Color Table
    - Why GIFs may not seem like "true color"
  - Uses **LZW Encoding** (Lempel-Ziv-Welch)
    - Create encodings based on strings of colors in image
    - Supplanted RLE for lossless compression
- Portable Network Graphics
  - Improved, non-patented replacement for GIF
  - Doesn't support animations

Color Table		
1	FF 00 00	
2	FF FF FF	
3	00 FF 00	

Hungary



Italy



Lossy Image Format: JPEG/JPG

- Joint Photographic Experts Group
  - Tradeoff between amount of compression and image quality
  - Areas of similar color are represented by a single shade
    - Based on quantization of discrete cosine transform (DCT) operation

Outline

- Images
- Compression
- **Strings**

String s= "hello"

char c = 'q'

s= "bye"

Strings

- A string is 0 or more characters "strung" together
  - Strings cannot be modified, but string variables can be reassigned
  - Individual characters can be accessed (not modified), numbered from left-to-right *starting at 0*

letters, numbers, symbols, spaces

- String literal: an unnamed string specified between double-quotes
  - *e.g.* `"hello"`, `"!@#$%^&*()_+ ?~"`, `"xoxo <3"`
  - `""` is known as the empty string (0 characters in it)

Using Strings

- Declaration: `String str;`
- Assignment: `str = "hello";`

  0 1 2

  dot notation

- Get character using `str.charAt(`~~1~~2`)` ⇒ 'l'
- Get length using `str.length()` 5

- Concatenation: join strings using '+' operator
  - *e.g.* `"hi " + "there"` gives you `"hi there"`

    addition w/ numbers
    concatenation w/ strings

- Conversion to string usually occurs *implicitly* "answer: 3"
  - Can also explicitly use `str()`

    "answer: " + 3
    "answer: " + str(3)

Strings vs. Arrays

- Strings are *sort of* like arrays of characters:

	Array	String
Declare	char[] chArray	String str
Initialize	chArray = {'h', 'i'}	str = "hi"
Get element	chArray[0] ⇒ 'h'	str.charAt(0) ⇒ 'h'
Get length	chArray.length ⇒ 2	str.length() ⇒ 2

Example: Recording User Input

- `keyPressed()` lets you read user input 1 character at a time

- Use a `String` variable to "store"
  - Add/append new characters using concatenation

Example: Recording User Input

- keyPressed() lets you read user input 1 character at a time
- Use a String variable to "store"
  - Add/append new characters using concatenation

```
String input = "";   // start with empty string

void draw() {
}

void keyPressed() {        convert char to string
  input = input + str(key);
  println("input = " + input);
}
```

string literal

concatenation

Word Guessing

- Learn to use text input & output
  - Player 1 enters a secret phrase
  - Player 2 tries to guess the secret phrase
  - Game tells you how many letters correct & # of attempts